ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.20, No.2, February 2009, pp.177−192
doi: 10.3724/SP.J.1001.2009.03365

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

动态需求跟踪方法及跟踪精度问题研究*

李 引[1,3+], 李 娟[1], 李明树[1,2]

[1](中国科学院 软件研究所 互联网软件技术实验室,北京 100190)

[2](中国科学院 软件研究所 计算机科学国家重点实验室,北京 100190)

[3](中国科学院 研究生院,北京 100049)

Research on Dynamic Requirement Traceability Method and Traces Precision

LI Yin[1,3+], LI Juan[1], LI Ming-Shu[1,2]

[1](Laboratory for Internet Software Technologies, Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

[2](State Key Laboratory of Computer Science, Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

[3](Graduate University, The Chinese Academy of Sciences, Beijing 100049, China)

+ Corresponding author: E-mail: liyin@itechs.iscas.ac.cn

Li Y, Li J, Li MS. Research on dynamic requirement traceability method and traces precision. *Journal of Software*, 2009,20(2):177−192. http://www.jos.org.cn/1000-9825/3365.htm

Abstract: Based on the current research on the dynamic requirement traceability (DRT), this paper investigates the trace precision problem in depth and proposes a solution—dynamic requirement traceability framework. This framework focuses on automatically establishing traces and combines the traditional requirement traceability activities with change proposal, impact analysis and change control, etc. It makes full use of the characteristics of artifacts and evolutional information of change in the iteration process so as to help establish the requirement traces with higher precision.

Key words: dynamic requirement traceability; requirement traceability; information retrieval; heuristic traceability rule; natural language processing; dynamic requirement traceability framework

摘 要: 在对动态需求跟踪现有研究综述的基础上,详细分析了动态需求跟踪面临的精度问题,并提出了解决方案——动态需求跟踪框架.该框架以自动化的需求跟踪为核心,组合了变更请求、影响分析、变更控制等活动,利用工作产品的特点和迭代过程中累积的变更信息来辅助建立需求跟踪关系,提高了需求跟踪关系的精度.

关键词: 动态需求跟踪;需求跟踪;信息检索;跟踪规则启发;自然语言处理;动态需求跟踪框架

中图法分类号: TP311 **文献标识码**: A

面向目标的需求工程的提出者 van Lamsweerde 在 2000 年的一篇需求工程的综述中说到,“软件需求在过去的 25 年的实践中被反复证实是一个确实存在的问题”[1].构建软件最困难的阶段是精确地决定要构建什么,所以,对于系统开发人员来说,最重要的一个活动是抽取和精化系统的需求[2].需求工程贯穿软件开发的整个生

* Supported by the National Natural Science Foundation of China under Grant Nos.60573082, 60803023 (国家自然科学基金); the National Basic Research Program of China under Grant No.2007CB310802 (国家重点基础研究发展计划(973))

Received 2007-08-07; Accepted 2008-04-15

命周期,“确定涉众和涉众的需求,并将其进行文档化,用来支持系统分析、交流协商和后期的实现”[3].“后期的实现”以需求为依据进行设计、开发、测试等活动.可以看出,需求与后期的实现之间有着紧密的联系,软件需求跟踪就是“一种描述和跟踪整个需求生命周期(包括前向和后向)的能力”[4].建立需求跟踪矩阵作为需求工程中一个指导性的标准过程而被越来越多的组织采用[4,5].它不但是 CMM/CMMI 2 级里面的一个核心活动[5−7],也是软件开发和维护标准 ISO9001 中条款 4.8 的一个必要活动[6].

需求跟踪辅助软件开发生命周期中很多活动的执行,并且为决策分析和权衡提供依据[5,8]:1) 对遗留系统中需求与设计和代码之间的关系进行分析,为新系统的开发提供范例[9];2) 可以作为确认和验证活动的一种手段,对系统功能进行确认和验证[7,10,11];3) 作为变更管理、风险评估的依据[12,13];4) 有助于理解后阶段工作产品为什么进行创建、修改和演化[14−16];5) 可以作为过程改进有价值的输入[17,18].一个全面的需求跟踪可以为最后开发的产品质量提供保证,对软件开发和维护提供有力支持,并且可以降低系统开发生命周期中的成本[5,19].实践表明,软件开发生命周期中“忽略了需求跟踪,或者是不充分和非结构化的需求跟踪关系将会导致系统质量的下降和反复的修改,从而增加系统开发的时间和成本.(由于缺乏有效的需求跟踪,以致)人员的流动会引起项目知识的缺失,从而导致作出错误的决策”[8].

近年来对需求跟踪的研究既有从技术手段角度着手(比如,关键阶段依赖[20]、需求工作产品超链接[21]、集成文档[22]等),又有从方法/框架角度着手(比如,需求引用模型[16]、需求交互管理[15]、方法驱动的跟踪捕获[13]),还有从工具对技术和方法提供支持的角度入手的(比如,商业工具 Rational Requisitepro[23]、DOORS[24]、CaliberRM[25]、Change Management & Requirement Tracing System[26]、原型工具 TOOR[27]、PRO-ART[28]等),但是在实践中我们发现,这些理论和工具对需求跟踪的支持并不是很好,需求跟踪没有完全发挥预期的作用[11],需求跟踪的核心问题依然没有得到解决,也就是缺乏一种有效的方法来降低建立和维护需求跟踪关系的成本.对于开发人员来说,建立和维护需求跟踪关系是一件耗时而且繁琐的事情;对于整个项目组来说,在需求跟踪上得到的收益小于付出的成本.特别是对于一个规模大、周期长的项目,面临着创建维护困难、容易出错、成本过高等问题[10−12,29].大量的实践经验表明,成本及效率问题是制约需求跟踪使用的根本原因.

动态需求跟踪从手动建立需求跟踪关系费时、费力且容易出错的问题入手,以自动化技术为手段辅助开发人员建立和维护需求跟踪关系,降低了成本.并且在需求跟踪的范围[30]、需求跟踪的粒度[31]等问题上,针对不同的项目特点,制定不同的跟踪策略[30],在成本、时间和可跟踪性等问题上进行了权衡,为需求跟踪面临的问题提出了一个较好的解决方案.

本文第 1 节对动态需求跟踪的研究现状进行分析和总结.第 2 节对动态需求跟踪面临的精度问题进行深入分析.第 3 节提出一个动态需求跟踪框架,以解决这个问题.

1 动态需求跟踪方法

Cleland-Huang 在 2005 年的需求工程大会上提出了“动态需求跟踪(dynamic requirement traceability)”[12]的说法,从命名上可以看出其与传统的需求跟踪的不同在于其“动态(dynamic)”.“动态”体现在需求跟踪中的一层意思是,需求跟踪关系的建立是自动化的[9,10,12,22,29,32−36];另一层意思是,需求跟踪关系是可以执行的,也就是说,在需求发生变更时可以根据跟踪关系进行变更提醒[37,38].可以看出,动态需求跟踪的核心是自动化的技术.

从跟踪方向上来说,需求跟踪可以建立需求与前期和后期的工作产品之间的跟踪关系;从跟踪内容上来说,可以建立需求与文本的工作产品和可执行的程序之间的跟踪关系.当把需求和工作产品都看成静态文本时,信息检索技术(information retrieval,简称 IR)、自然语言处理、人工智能、人机交互等技术[30]可以自动建立需求与工作产品之间的跟踪关系;当把需求或者场景(scenario)看作静态文本时,使用监测程序执行[39,40]方面的技术能够自动建立与执行程序中功能之间的跟踪关系.因此,现阶段对动态需求跟踪的研究主要是将其他领域的技术应用到需求跟踪中,包括信息检索技术、自然语言处理、运行时分析、事件发布/订阅等.下面将按照这几种技术来对动态需求跟踪的研究进行概述.

1.1 基于信息检索(IR)的动态模式

基于信息检索的动态模式利用 IR 模型计算需求文本与工作产品文本之间的相似度,按照相似度的大小进行排列,通过设定一定的阈值就可以筛选得到需求所跟踪的工作产品.需求与工作产品之间的相似度越大,它们之间的跟踪关系就越有可能是正确的,反之则越有可能是错误的.IR 技术的优点在于,通过对本文进行分析,能够自动地建立关系,而不需要人的介入,这对于传统需求跟踪活动所面临的维护困难、容易出错、成本高等问题,是一个很好的解决方案.该模式通过需求与工作产品中标引词(term)的匹配计算出相似度,用其代替对语义的理解,优点在于简单.但其缺点也是显而易见的,如果标引词无法匹配,那么即使两个标引词描述的意义相同或者相近,计算得出的相似度也会很低,可能会造成相似度过低而无法建立跟踪关系,与实际的语义相悖.在一个周期长、规模大的项目中,需求、设计和编码通常都是由不同的人来完成的,采用不同的标引词来表示相同语义的情况相当普遍.而且在软件开发的各个阶段,由于描述的领域不同(比如需求是对问题领域的描述,编码是对解决方案领域的描述),采用的描述语言一般会有所不同,因此采用不同的词对同一个概念进行表示是可以理解的.从以上分析可以看出,采用 IR 技术来解决需求跟踪的一个关键问题是如何解决一词多义和同义 多词.

下面首先对 IR 技术作一个简单的介绍,然后根据将 IR 技术应用到需求跟踪领域的 3 种情况(直接应用、根据项目特点修正应用和根据工作产品特点修正应用)来论述现有的研究,最后是根据项目特点综合应用不同的跟踪策略并进行结果度量.

1.1.1 IR 简介

信息检索是指用户为处理解决各种问题而查找、识别、获取相关的事实、数据、文献的活动及过程[41],简单来说,信息检索就是研究由用户给定的查询返回相关文档问题的领域[29].

在大量的 IR 方法中,基于标引词(keyword-based)的信息获取是最常用的方法[29].待查询的文档被认为是具有不同重要程度标引词的组合,用户查询也被认为是标引词的组合,与查询相关的文档通过比较它们之间的标引词来确定.

用户查询和文档之间的相关性通过 IR 相似度模型来计算,主要包含 3 大基本模型:布尔模型(Boolean model)、向量空间模型(vector space IR,简称 VSIR)和概率模型(probability IR,简称 PIR).

1) 布尔模型[41].这是最简单的模型,它将一个标引词是否在文献中出现表示为 1 或者 0.

2) 向量空间模型[41].该模型扩展了布尔模型,将文献集合中所包含的标引词看成向量,代表空间中的一个维度,这些标引词集合就构成了一个空间.文献集合中的任何一个文献都可以用空间中的一个向量来表示,利用查询和文献的向量积就可以计算它们之间的相似度.假设集合中有 n 个不同的标引词,则文献 D_i 可以表示为 $D_i=(d_{i1},d_{i2},\ldots,d_{in})=\sum_{j=1}^{n}d_{ij}\vec{t}_j$[41],其中,$d_{ij}$ 表示标引词 j 在文献 i 中的权重(weight);查询 q 可以表示为 $q=(q_1,q_2,\ldots,q_n)=\sum_{j=1}^{n}q_j\vec{t}_j$[41],其中,$q_j$ 表示标引词 j 在查询中的权重.运用向量积运算可以得到文献向量 D_i 与查询向量 q 之间的相似度:$Similarity(D_i,q)=D_i.q=\sum_{k,j=1}^{n}d_{ik}q_j(\vec{t}_j\vec{t}_k)$[41].同一个标引词在不同的查询和文献中具有不同的权重,权重值越大则认为该标引词越能代表这篇文献,权重值越小则认为该标引词越不能代表这篇文献.Tf-idf 是向量模型中比较常用的标引词权重的计算方式[41],综合考虑了标引词的词频和逆文献频率两个因素,比较全面地反映了标引词在整个文献空间中的重要性.词频用 tf 表示,是标引词 i 在文献中出现的频率,其计算方式为 $\dfrac{freq(t_i,d)}{\sum_j^m freq(t_j,d)}$,$m$ 表示文献中包含的标引词的个数,$freq(t_i,d)$表示在文献 d 中标引词 i 出现的次数;文献频率 df 是包含该标引词的文献数与所有文献数的比值,逆文献频率 idf 的一种计算方式是 $\log_2\left(\dfrac{n}{df}\right)$,$n$ 表示文献的个数.因此,一个标引词 i 的权重表示为

$$w_i=tf_i\times idf_i=\frac{freq(t_i,d)}{\sum_j^m freq(t_j,d)}\times\log_2\left(\frac{n}{df_i}\right).$$

3) 概率模型[41].该模型将 IR 问题用概率加以解释,认为文献与查询之间的相关性是通过标引词作为中间桥梁来计算的.任何一个查询和文献都可以表示为包含一定标引词的命题,它们之间的相似度按照条件概率进行解释:当存在查询时,存在文献 i(i=1,2,…,n,n 为文献数)的概率.其相似度计算公式如下:

$$Similarity(D_i,q)=\Pr(D_i\mid q)=\frac{\Pr(q\mid D_i)\Pr(D_i)}{\Pr(q)}.$$

其中,D_i 表示文献 i,q 表示查询.也就是说,查询 q 与文献 D_i 之间的相似度等于当 q 存在时 D_i 也存在的概率.

计算出相似度之后,根据相似度大小进行排列,值越大,说明该文献越能反映用户的期望;值越小,则越不能反映用户期望.用户的期望与实际返回的结果是有一定差距的,可以采用查准率(precision)和查全率(recall)对结果进行度量.按照文献[33]中的定义,查准率是查出的正确的文献数与查出的所有文献数的比值,而查全率是查出的正确的文献数与所有正确的文献数的比值.

1.1.2 直接应用 IR 技术

Antonial 等人首先将 IR 技术应用于恢复代码到需求的跟踪关系中[32,33].需求文档采用自然语言进行描述,假设代码内的类名、函数签名、变量等遵循有意义的命名方式.建立从代码到需求的跟踪关系就可以看成是以代码为查询条件、以需求为文献的一个 IR 处理过程,通过对需求文档和代码用 IR 技术处理就可以建立跟踪关系.由于自然语言表达的多样性,需求文档和代码需要经过文档标准化的预处理的过程,包括将大写字母转化为小写字母、去掉停顿词、生成词根等操作.这个过程的关键是尽量将自然语言表达上的“噪音”去掉,使具有相同意义的词句能够更好地匹配.然后,采用 IR 中的向量空间模型和概率模型来计算相似度.这两种模型被分别用来验证最后建立的跟踪关系的正确性.实验结果表明,两种模型都适用于恢复从代码到需求的跟踪关系,但是概率模型达到的效果更好[33].

在 Antonial 直接将 IR 模型引入到恢复代码与需求之间的跟踪关系的研究之后,Andrian 针对其查全率和查准率较低的问题,提出了采用隐含语义标引(latent semantic indexing,简称 LSI)[42,43]的方法来计算相似度[9].从 IR 技术的相似度计算公式可以看出,造成查全率和查准率低的一个根本原因在于语法上的字词匹配无法反映真实的语义.隐含语义标引的方法不依赖于标引词的匹配来计算相似度,而是通过分析文本之间隐含的“语义结构”来确定关联度.该方法基于向量空间基本模型,不但能够捕捉单个字词的语义,而且还能捕捉句子、段落以及整个文献的语义.LSI 方法分析了某个词出现或者不出现的语境,通过这个语境设定的约束就决定与其他具有类似语境的词的相似度.因此,在检索中使用的是基于这个“潜在”语境的描述,而不是原来字面上的字词描述.需求与代码之间的相似度是由它们所采用的全部标引词的使用方式决定的,也即取决于它们中所包含的标引词的意义在多大程度上是一致的.

当把需求和代码表示成“标引词-文档”向量之后,采用奇异值分解的方法就可以创建 LSI 的子空间,也就是 $X=T_0S_0{D_0}^T$,其中,X 是 $t\times d$ 阶的标引词-文献矩阵,T_0 和 D_0 各列都是正交的.通过这样的奇异值分解,就将原有的向量空间分解为 T_0 代表的“标引词”矩阵和 D_0 代表的“文献”矩阵,S_0 中的对角线上是矩阵 X 的各奇异值.新的文本向量可以通过投影到 LSI 子空间的方式来获得.奇异值分解的一个优点就是允许用一个维数较低的矩阵作原来矩阵的最优近似,将维数降低的目的是需要将原来的 m(标引词的个数)个标引词表示的相同或者相近语义,分配到各个“概念”分量上.将 S_0 中的奇异值按照重要性筛选出最大的 k 个,同时也将 T_0 和 D_0 的最右边的列去除一部分变成 k 列,形成新的矩阵 $X'=T_kS_k{D_k}^T$,在将文献集合投影到这个 k 维空间时,与原来文献的距离平方差最小.因此,最后计算需求与代码之间的相似度是通过各自近似的矩阵 X'进行矩阵运算而得出的[9].

文献[9]采用的实验数据集与文献[33]完全相同,从实验结果分析来看,LSI 的查全率和查准率均高于文献[33].而且文献[9]的限制条件不需要需求与代码在同一概念的描述上都采用相同的词,其条件是一些词的使用存在大量相同或者类似的语境,即一些词的出现能够强烈地暗示其他词的出现.通过 LSI 的方式还可以去除由人为原因引入的匹配问题,比如拼写错误,这些干扰可以在用奇异值分解的过程中被过滤掉.但是,LSI 采用的奇异值分解的方式可能会模糊一些能够明显标识一篇文献的标引词的作用,而且对于缺乏相同语境的标引词来说,不能通过分解的方式将其分配到概念子空间上,所以也存在精度问题[41].另外,LSI 存在数值存储空间较

大、运算量较大的问题,对于网络搜索这类应用来说是非常致命的,但是文献[9]引入 LSI 的一个初衷就是考虑到需求跟踪领域中不可能出现海量的数据,而且时间也不是所考虑的最主要的因素.因此,Andrian 利用了需求跟踪领域的特殊情况,采用适当的方法对计算方式进行优化,从而得到更高精度的跟踪关系.

1.1.3 根据项目特点修正应用

Hayes 在 Antonial,Andrian 等人的研究基础上,用向量空间模型作为基本模型,根据项目特点(如项目数据字典)提出了辅助手段,以提高跟踪关系的精度.

文献[29]在经典的 tf-idf 权重模型基础上,引入 key-phrase 来解决同义不同形的词的匹配问题.主要是通过分析项目数据词典,明确需求跟踪关系的建立过程中会遇到的词汇,这样一些词汇就会获得较高的权重.另外一个辅助手段是在 key-phrase 的基础上,对传统的向量空间模型相似度的计算公式进行非正交的修改.传统的向量空间模型认为标引词所表达的意义是完全正交的[41,44],而在实际中,标引词所表达的意思大多是重叠甚至重合的,因此最后的结果必然存在偏差.文献[29]采用标引词相似度词典来记录标引词之间的相似度,每一个词典项为一个三元组(k_i,k_j,α_{ij}),其中,k_i 和 k_j 为标引词;α_{ij} 表示它们之间的相似度,取值为[0,1].最后的相似度计算公式经过修正表示为

$$Similarity_T(D_i,q)=\frac{\sum_{i=1}^{n}w_i\times q_i+\sum_{(k_i,k_j,\alpha_{ij})\in T}\alpha_{ij}\times(w_i\times q_j+w_j\times q_i)}{\sqrt{\sum_{i=1}^{n}w_i^2\times\sum_{i=1}^{n}q_i^2}}.$$

实验结果表明,词典方法的精度远高于采用基本向量模型的结果,在查全率为 85%左右时,查准率能够达到 40%.文献[10]中引入了相关性反馈来改善跟踪关系的结果,其原理是通过用户提供的正确的跟踪关系和错误的跟踪关系对查询向量中的标引词的权重进行修正.标准的 Rochio 方法计算如下:

$$q_{new}=\alpha q+\left(\frac{\beta}{r}\sum_{d_j\in D_r}d_j\right)-\left(\frac{\gamma}{s}\sum_{d_k\in D_{irr}}d_k\right),$$

表示新的查询向量是由原有查询向量加上用户反馈的正确的跟踪关系集合中的文档的平均向量,并减去用户反馈的错误的跟踪关系集合中的文档的平均向量而得出的.增加正确跟踪关系的向量提高了查全率,而减去错误跟踪关系的向量则提高了查准率.

文献[29]中实验了几种扩展策略:tf-idf,tf-idf 加上 key-pharse,tf-idf 加上标引词相似度词典以及反馈方法.文献[11]中将这几种方法与文献[9]中采用 LSI 的方法进行了结果分析,发现 tf-idf 加上标引词相似度词典和反馈方法的效果要好于 LSI 加上反馈的方法.这是因为采用了标引词相似度词典的向量空间模型实际上显式地处理了标引词之间的语义对应关系,人为建立的标引词相似度词典的精度必然比通过奇异值分解得出的 LSI 概念子空间的精度要高.如果两个标引词之间的出现没有联系,也就是说,相同语境的情况不多甚至没有,LSI 就无法达到想要的结果.

文献[34]在向量空间模型的基础上引入了轴反馈技术[41,44]和标引词词典[29]两种辅助方法来建立需求到几种 UML 设计模型和代码的跟踪关系.研究发现,建立需求到设计的跟踪关系的精度高于建立需求到代码的跟踪关系的精度.对于采用模型驱动(model driven architecture,简称 MDA)的方法来说,从设计到代码存在很多隐式的联系,因此,利用 UML 设计模型作为中间桥梁可能会是一种比较有效的建立需求到代码的跟踪关系的方法.

之前的方法中相似度的计算是通过匹配查询与文献之间的标引词来进行的,由于同一个标引词可能表示不同的概念,而且同一个概念也可能由不同的标引词来表示,因此,获取的结果就可能包含不相关的文献从而降低了精度.文献[45]引入了分词技术,采用 POS(part-of-speech)标签来分析需求文档和工作产品,并且利用了项目词汇表,根据重要程度来设置不同标引词的权重,能够更好地提高查准率.文中通过实验证明,利用分词产生的名词短语在需求文本和工作产品之间同时出现的概率值来修正 IR 的相似度计算公式,可以在保持查全率的情况下,提高查准率.

1.1.4 根据工作产品特点修正应用

Cleland-Huang 等人利用工作产品所具有的特点(如工作产品的层次结构信息)来修正 IR 技术,与 Hayes 等

人利用项目的特点来修正IR技术相比,其分析粒度更细,IR技术被越来越恰当地裁剪出适合需求跟踪这个特定领域的方法.

文献[12]中采用的第 1 个增强策略是利用需求和工作产品的层次结构信息(hierarchical information).它所基于的一个假设是,如果需求的祖先需求和工作产品的祖先工作产品已经建立跟踪关系,该需求和工作产品建立跟踪关系的可能性就大大提升了.因此,可以用这种层次结构信息对概率模型的计算公式进行修正,从而提高候选跟踪关系的精度.第2个增强策略是利用了IR中的聚类技术(clustering of artifacts),如果一篇文档能够与一个查询相关联,那么这篇文档所属的聚类中的所有其他文档也会有较高的概率与这个查询建立关联关系.这两种策略的实验效果相差不大,所获取的正确关系有很多是重复的.究其原因,是它们都认为相近的文档就会有类似的跟踪关系存在.因此,聚类的文档集合可能会与某个根文档及其以下的文档组合的文档集合有很多重叠,所以才会有重叠的跟踪关系存在.最后一个策略是人工进行手动剪枝(tree pruning)操作.如果确定了某个需求与某个工作产品不应该存在跟踪关系,那么,即使计算出来的相似度高于阈值,也可以人工地约束这个需求所属的集合(聚类或者层次结构的)中的需求不能与工作产品所属的集合(聚类或者层次结构的)中的工作产品建立跟踪关系.实验结果表明,每一种策略都不是万全的,对于不同的项目数据来说,会有不同精度的提升.因此,根据项目特点来选择适当的策略是很有必要的.第一种策略已经考虑到了需求跟踪这个领域的特点,利用层次结构信息来辅助 IR 方法,但也仅考虑了祖先子孙节点包含关系,对于结构上的一些更加丰富多样的关系(比如依赖关系)则没有考虑,因此,其增强策略并不完整,还可以进一步深化扩展.

在以往的研究中,由于非功能性需求一般是对一个系统的整体约束,而且它们之间存在着纠缠不清的相关性,导致很难进行追踪[46−48].而事实上,非功能性需求的实现和变更管理在很大程度上会影响一个系统的成功与否[35].文献[35,49]引入了需求工程领域中的软目标互依赖图(softgoal interdependency graph,简称 SIG),与IR技术组合运用,提出了“以目标为中心的跟踪(goal-centric traceability,简称 GCT)”的方法,对非功能性需求(non-functional requirement,简称 NFR)的依赖和变更进行了管理.该方法分为4个部分:首先是用软目标互依赖图来捕获系统 NFR 相互之间的依赖关系以及 NFR 对应的各个实现(operationalization);然后利用 IR 方法在软目标互依赖图中的元素与 UML 设计类图之间建立动态跟踪关系;当 NFR 发生变更时,通过软目标互依赖图和 IR 建立的跟踪关系就可以确定哪些 UML 设计类图需要发生更改;最后对这些影响进行分析并作出最后的决策.整个方法基于软目标互依赖图建立 NFR 与实现之间的关系,并结合 IR 建立的实现与类之间的跟踪关系,在对 NFR 进行变更时能够立刻获得可能会发生变更的类图.但是,IR 方法所特有的缺陷导致了精度是其中一个不可避免的问题.

1.1.5 跟踪策略与度量

需求跟踪具有重要作用已经得到认可,越来越多的需求跟踪手段也已被提出来,但如何得到较高的投资回报率(return on investigation,简称 ROI)还是很值得探讨的[50].文献[30]从项目级的角度入手,对不同的特定项目应该采取不同的跟踪关系策略进行了讨论.作者认为,一个项目的特点应该对选择使用的跟踪关系技术起决定性作用,选择和衡量一个跟踪策略应该从自动化程度、正确率、查全率这 3 个方面进行综合考虑,以一个适当的 ROI 来达到所期待的目标.对不同的需求跟踪失败的风险可以从发生变更的概率和发生变更之后的影响两个方面来度量,其计算出来的数值就可以用来指导采用不同的跟踪策略,比如对于 Substantial 级别,建议对 50%的功能性需求采用手动方式建立跟踪关系以提高查准率,对于少于5%的NFR可以采用基于事件的需求跟踪方式,对于45%左右的需求可以采用精度不太高的基于IR的动态需求跟踪方式.采用这样的量化方法之后,就可以更好地应用跟踪策略,将需求跟踪的价值最大化.

文献[10]分析了采用IR技术建立动态需求跟踪关系的工具所应满足的需求,包括可信性、可分辨性和持久性.可信性主要是对精度、伸缩性和有效性的要求;可分辨性是指分析人员应该能够从工具生成的候选跟踪关系中选择出正确的跟踪关系;持久性是针对建立跟踪关系并接受用户反馈之后,能否使以后的候选跟踪关系精度有所提高而提出的要求.针对这样一些需求,Hayes 等人提出了称为第二度量的公式[10],对是否满足需求进行评判.除了标准的查全率和查准率以外,F-measure 综和了查全率和查准率,以一个更加全面的方式来考虑精度:

$$f=\frac{1+b^2}{\frac{b^2}{recall}+\frac{1}{precision}};$$

Selectivity 公式从用户对候选跟踪关系进行再分析的工作量上进行度量:

$$selectivity=\frac{\sum_{q\in H}n_q}{M\times N}.$$

其中,$M\times N$ 表示候选的跟踪关系;Lag 公式对一个候选跟踪关系中正确的跟踪和错误的跟踪关系的可区分度进行刻画,lag 值越小,正确的跟踪关系和错误的跟踪关系就越容易区分,候选跟踪关系的 lag 值计算为

$$lag=\frac{\sum_{(q,d)\in T}lag(q,d)}{|T|},$$

其中,T 表示所有正确的跟踪关系集合,$lag(q,d)$表示所有错误的跟踪关系 $lag(q,d')$相似度大于 $lag(q,d)$的数目.检验动态需求跟踪工具是否满足需求可以由这些度量公式来验证,见表 1.

Table 1 Measures classification

表 1 度量分类

Requirement	Measures
Believability::Accuracy	precision, recall, *F*-measure
Believability::Scalability	precision, recall, *F*-measure
Believability::Utility	selectivity
Discernability	lag
Endurability	selectivity

文献[51]对现有的采用 IR 的动态需求跟踪的实验进行了分析比较,并提出了一个实验框架来指导构造实验和比较研究成果.

1.2 其他模式

1.2.1 基于跟踪规则启发的模式[14,36]

Spanoudakis 提出了一种启发式的跟踪方法[14,36],称为基于规则的动态需求跟踪.在该研究中,Spanoudakis 将动态需求跟踪分为两种情况,其一是需求文档或用例与工作产品之间的跟踪关系;另一种是需求文档以及用例相互之间的跟踪关系,这种关系是在这些文档与工作产品之间的跟踪关系的基础上建立的.对于这两种不同情况,定义了不同的匹配规则.Spanoudakis 还将所有的跟踪关系按照其逻辑上的关联分为 4 大类,使得工作人员可以据此清晰地了解具有跟踪关系的项目之间彼此驱动的先后顺序和逻辑关系.在具体实现过程中使用了一系列自然语言处理的方法,首先对需求文档和用例分别进行语法分析,标出句中词语的词性以及在句子中所构成的成分;然后将各个句子按词拆开,标注其成分标记,存储成 XML 的格式;最后根据前面提到的规则,将各个 XML 文件的内容进行跟踪关系的匹配.该方法的优点在于可以建立任何文档之间的关系,缺点在于需要针对不同的项目定制不同的跟踪关系匹配的规则.

1.2.2 基于运行时分析的模式[39,40]

文献[39,40]采用对系统运行时的信息进行监控的方法,发现开发模块(代码类、数据流图等)与系统的实际实现和运行模块之间的依赖关系,然后利用这些依赖关系生成跟踪关系.该方法的关键在于使用 Rational PureCoverage(目前升级为 PurifyPlus[52])工具来获取运行时的信息,并在运行的程序与静态的模块之间建立关系.运行时捕捉的信息称为“脚印(footprint)”,用来记录测试场景在执行时所使用的类及使用次数.

整个方法包含 4 个主要的处理过程:建立假设依赖关系(hypothesizing)、原子化(atomizing)、泛化推理(generalizing)和精化推理(refining).

(1) 建立假设依赖关系.主要是由人手工建立一些显而易见的模块之间的依赖关系,这些关系往往是从文档中归纳出来的.这一步建立的关系并不能保证完全正确和完整,它只是后面推理的一个开始,如果其中的关系有错误,可以在后面的推理过程中检查出来.

(2) 原子化操作.利用 PureCoverage 工具来获取运行时信息;并建立一个“脚印图(footprint graph)”,对场景

之间相同的原子模块进行刻画.当多个模块共享同一段代码时,这些模块之间就存在关联关系,利用这个原理就可以发掘出模块之间的依赖关系.

(3)-(4) 这两步根据前面生成的脚印图,利用节点之间的泛化和精化关系就可以推理出更多的场景与模块、模块与模块等之间的关系.

这种方法与基于 IR 和跟踪规则启发的方式有很大的不同,它是基于模型监测技术来建立动态执行程序与静态文档之间的关系,与其他方法相比,它能够更精确地在程序的行为与静态文档之间建立关系,甚至可以建立非功能性需求与代码之间的关系[40].但是其不足也是比较明显的,它需要一个能够运行的系统和定义好的场景才能建立关系,这属于事后(after-the-fact)建立跟踪关系的方法.

1.2.3 基于事件的发布/订阅模式[37,38]

基于 IR 和基于规则的动态需求跟踪技术是对需求和工作产品进行静态分析之后建立跟踪关系.在需求或者工作产品发生变更之后,需要重新计算或者匹配才能确定新的跟踪关系.如果把它们看成是静态的模型,那么基于事件的需求跟踪模型就是动态的、可执行的模型.

基于发布/订阅事件的需求跟踪模型的关注点与前两类方法有所不同.前两类方法是从访问/表现跟踪信息的角度来解决问题,而该方法是从组织/维护跟踪信息[38]的角度来解决问题.在时间紧、任务重的项目环境中,开发人员可能会忽略对变更的响应,以至于工作产品的生产和维护与需求不一致,基于事件的需求跟踪模型采用了发布/订阅的事件模式对需求与工作产品进行松耦合管理.开发人员将工作产品订阅到需求上,当需求发生变更时,系统会以通知的形式提醒开发人员需要进行更新的模块.需求的变更包含:创建、删除、修改、合并、精化、分解、替换等类型,每一种类型都可以由链接之间的基本操作组合而成.因此,每一个通知消息都属于一种类型.根据需求的重要程度和消息发布的时间,通知消息以不同的表示形式来反映其重要程度.整个模型不但可以对直接的跟踪关系中需求的变更发出通知,还可以对间接的多层跟踪关系路径上需求或者工作产品的变更发出通知.在通知发出的时间控制上,既可以采用暂时缓存通知的“延迟通知”方式,又可以采用立即发送通知的“悲观通知”方式.在追踪关系的维度上,既可以在需求与工作产品之间建立跟踪关系,又可以在不同版本的需求之间建立跟踪关系,从而在空间和时间上对需求的变更进行全面的跟踪.

整个模型的实现从整体架构上来看分为 3 个部分:需求管理模块、事件管理模块和订阅管理模块.需求管理模块对需求的层次结构进行组织,对需求的变更进行记录.每一次的变更都可以用原子操作来进行描述,并可以将其发布到事件管理模块.现有成熟的商业需求管理工具都提供了需求变更的详细记录,通过分析这些记录可以构造出相应的通知消息.事件管理模块主要是管理订阅消息,接收需求变更事件并根据事件创建通知并分发.这里可以采用一些策略对接收和分发进行控制.订阅管理模块根据接收到的通知提示订阅者需要对变更进行处理.

这种需求跟踪模型的优点是将变更作为事件发布到订阅处,起到了提醒、督促的作用.由订阅者对复杂的变更情况进行手动处理,增量式地建立下一个时刻的需求跟踪关系,以这样的方式建立的需求跟踪关系的精度会很高.这种模式的问题在于,手动地处理变更虽然将大量的需求跟踪问题分解为少量、多次的处理,但这依然是一个时间耗费大、容易出错的过程.该需求跟踪模型的初始化同样存在手动建立需求跟踪关系的一切问题.

1.3 方法的比较

动态需求跟踪方法从应用上来说,大体分为两类,一类是用来建立需求跟踪关系,比如基于 IR 技术、基于启发式规则和运行时分析;另一类是执行需求跟踪关系,比如基于事件的引擎.方法的对比见表 2.

基于 IR 技术和启发规则的方式对静态的文本进行分析,根据各自定义的近似跟踪关系语义来建立关系.基于运行时分析的模式则另辟蹊径,对运行系统进行监控并记录下运行功能与静态文本之间的对应关系,通过进一步推理,建立系统功能与静态文本的跟踪关系.

采用 IR 技术的动态需求跟踪都比较简单,文本在进行简单的自动化处理之后,就可以用来计算文本之间的相似度.这种方法的应用基本上不需要用户的介入(除了最后对候选跟踪关系进行筛选以外),而且能够对任何形式的文本进行分析和处理,因此其具有的特点是自动化程度高,对文本的约束弱,但精度问题也是其所面临的

最大挑战.

Table 2 Method comparison

表 2 方法的比较

Category	Type	Object/Constraint of trace	Usage	Characteristic
IR	VSIR[10,11]	Arbitrary	N/A	Basic model of IR, keyword matching
	PIR[12]	Arbitrary	N/A	Basic model of IR, keyword matching
	LSI[9]	Arbitrary	N/A	Extended VSIR, text context matching
	Thesaurus[11]	Arbitrary	N/A	Define similarity coefficient between keywords
	Hierarchical modeling[12]	Between hierarchical requirement and artifact	N/A	Hierarchical information of requirement and artifact is useful to filter the candidate traces
	Logical clustering of artifacts[12]	Cluster the artifacts	N/A	Clustering is helpful to recall
	Goal-Centric traceability[35]	Between NFR and design pattern	N/A	Synthesize SIG and IR
Heuristic rule-based generation[36]	N/A	Requirement and use case specification, and analysis object model	N/A	Extract heuristic rule from natural language document
Run-Time analysis[39,40]	N/A	Test scenario, models and UML	After building executable program	Establish trace between running behavior and static document
Event-Based traceability[38]	N/A	Arbitrary	Real-Time monitoring	Requirement and artifact are linked through publish-subscribe relationship

启发式规则需要手动建立需求规约、用例和分析对象模型之间的规则,然后通过这些规则来建立关系.对于不同的项目,需要对系统进行整体分析之后再开发不同的启发规则,这也是一件耗时、耗力的事情.其优点在于能够精确定制需求之间的相关性规则,从而自动建立需求之间的相关性.

采用运行时分析的方法与以上两种方法相比能够达到更高的精度,因为它在运行程序的行为与文档之间建立跟踪关系,这种在动态行为与静态需求之间的关系能够更好地用来进行验证和确认活动.该方法还具有能够追踪非功能性需求、帮助发现冲突需求和理解跟踪关系的强度这 3 个特点.其缺点在于,需要一个已经能够运行的程序/构件,因此对于系统还没有可以运行的构件时,跟踪关系是无法建立起来的.

1.4 动态需求跟踪工具

在这几类动态需求跟踪方法中,由于基于信息检索的动态需求跟踪具有自动化程度高的特点,因此已有一些相对较成熟的工具出现,Poirot[53]和 RETRO[54]就是其中的代表.

Poirot[53]是一种基于 Web 的动态需求跟踪工具,利用 IR 中的概率模型来计算相似度.其最大的特点在于能够利用信息检索技术自动地在第三方的 CASE 工具(包括需求工具 DOORS、设计工具 Rational Rose、代码库等)产生的工作产品之间建立跟踪关系.由第三方的 CASE 工具产生的工作产品被适配器(adapter)转换成特定格式要求的 XML 文档,输入 Poirot 并进行索引操作.用户通过 Web 界面可以进行两种形式的查询:一种是获取以需求、UML 设计或者代码为源头的跟踪关系的结果;一种是自由输入文本,查询到对应的工作产品.该工具平台已经搭建并测试了来自西门子公司的项目.

RETRO(requirements tracing on target)[54]基于向量模型,并综合了反馈技术来改善跟踪关系.需求文档和工作产品被索引和存储,用户查询指定需求对应的工作产品,并根据返回的跟踪关系进行筛选,系统根据反馈的结果调整查询向量从而产生修正之后的跟踪关系结果.该工具最新版本为 2.5,包括全自动跟踪模式和手动跟踪并浏览模式,提供了全部跟踪、单项跟踪、正确/错误跟踪关系反馈等功能.

2 跟踪精度问题分析

动态需求跟踪关系是为了降低人力资源的耗费和减少手动建立跟踪关系出现错误而提出来的,其最大的优势在于自动化处理能够提高跟踪效率,降低跟踪成本.自动化处理采取某种近似的方式来模拟人的语义理解,但是产生的跟踪关系一般会存在错误,并可能漏掉部分正确的关系.尤其对于精确性和安全性要求较高的项目,正确无误的跟踪关系为系统的演化提供了可靠的保证,是项目成功的一个前提.因此,跟踪精度问题是现有研究

中的一个重点.

采用 IR 技术的方式通过分析文本之间的相似度来建立跟踪关系,具有自动化程度高的特点,因此现阶段的研究大多集中在采用 IR 技术建立需求跟踪关系上.下面基于我们在以需求为中心的变更影响分析(requirement based change impact analysis)研究中[55]遇到的查准率和查全率低下的情况,对跟踪精度的问题进行深入分析.

2.1 查准率与查全率分析

在信息检索技术中,查准率和查全率是衡量检索结果的两项指标.简单地通过降低筛选阈值提高查全率的方法,自然也就降低了查准率;反过来,提高筛选阈值虽然提高了查准率,但却降低了查全率.调整阈值的方法只是简单地套用 IR 技术中的筛选方式,并没有将需求跟踪的特点引入其中.下面将从查准率和查全率的角度进行分析,从而发现精度问题的根源所在.

2.1.1 查准率

从查准率的定义可以看出,提高查准率的一种很明显的方式就是减少错误跟踪关系.错误跟踪关系是与正确跟踪关系相对的,是指不该发生的跟踪关系由于计算出来的相似度大于筛选阈值,从而引入到跟踪关系中.

在我们的研究中发现,相似度计算中采用的经典的 tf-idf 词权重模式在某些情况下会造成查准率降低,它表示为

$$w_i=tf_i\times idf_i=\frac{freq(w_i,d)}{\sum_j^m freq(w_j,d)}\times\log_2\left(\frac{n}{df_i}\right).$$

从计算公式我们可以看出:1) 词在文献中出现的次数越多,*tf* 的值越大;2) 词在越少的文献中出现,*idf* 就越大.在包含相同数量的标引词 *x* 的文献集合中,一个文献中包含的标引词总数越少,*x* 的 *tf* 就会越大;同时,当 *x* 在文献中出现的次数越少(专指程度高),*x* 的 *idf* 就越大.比如,在我们研究的案例系统需求管理工具[55]中,存在类 *BaseDbOperation* 封装了对数据库的原子操作,按照规则可以将其拆分为 *Base*,*Db* 和 *Operation* 三个词.由于其包含较少数量的标引词(3 个),导致了标引词 *Base* 在权重公式 *tf* 部分的权值较大.同时,标引词 *Base* 在需求 *base data management* 和 *baseline management* 中出现,最后计算出的类 *BaseDbOperation* 与这两个需求之间的相似度大于预设阈值,从而产生错误的跟踪关系.

从上面的分析可以看出,在 IR 模型中考虑的是词的非语义信息的重要程度,它只是根据词出现的次数来计算.而需求跟踪有自己的特点,不应该一成不变地套用这种模式,而应该将语义信息引入,通过细分之后的需求跟踪对象的特点来对 tf-idf 的权重模型进行修正.

2.1.2 查全率

需求和工作产品一般都是采用两个具有大致相同概念范畴的词表来进行构建,同一个概念在两个词表中就有可能采用不同的词进行表示.因此,采用 IR 模型计算需求与工作产品之间的相似度,就会因为具有相同/相似概念标引词的不匹配而造成相似度的降低.这可以说是导致查全率过低的一个根本原因.从具体的表现形式上来说,一般有两种情况会造成相关的需求和代码不匹配:1) 需求文档不采用缩写,而代码采用缩写;2) 需求文档和代码采用不同的标引词来表示同一个意思.在我们的案例系统需求管理工具中,这两种情况都有出现.比如,文档中对需求的描述采用词 *requirement*,而在代码中出于简洁性将其缩写为 *req*;在需求中描写权限采用 *purview*,而在代码中却采用 *authority*.针对第 1 种情况,我们采用字符串编辑距离来计算匹配完整的词和缩写所需要编辑的次数,如果编辑次数小于设定的阈值,则认为这两个词是同一个词.对于第 2 种情况,我们采用基于认知语言学的英语词典 WordNet[56](按照词的意义组织成一个网络)来自动计算词之间的距离,在距离小于一定阈值时,就可以认为具有相同词义.

在以往的研究中,为了提高查全率,大多采用降低筛选阈值的方式,却引入了更多的误链接,致使查准率降低[33−35].如果计算出来的需求与工作产品的相似度小于筛选阈值,而它们之间实际上应该有跟踪关系,那么,在不改变阈值的情况下,怎样才能将其引入呢?文献[34]中的第 2 种策略对具有层次结构的制品的相似度计算方式进行了修正,从而避免建立误链接,也加强了正链接.另外一种方式是直接针对同义标引词不匹配的问题,通

过 LSI 或者建立标引词相似度词典来获取更全面、更准确的跟踪关系.这两种方式都是利用文本的特点,降低了单独采用字词匹配所带来的影响,这是提高查全率的一种关键方法.

2.2 精度问题根源

从对查准率和查全率的分析来看,动态需求跟踪精度问题的根源在于需求与工作产品之间的“实现”语义是不能被机器所理解的,为了利用机器的自动化特性,只有将这种“实现”语义用描写需求与工作产品的文本之间的“相似度”来代替,从而采用 IR 技术,自动地建立需求跟踪关系.由于设计与代码是最重要的工作产品之一,设计与代码之间存在着一定的对应关系,所以,下面以建立需求到代码的跟踪关系为例,对精度问题的根源进行详细分析.图 1 为建立需求到代码的跟踪关系的示意图.

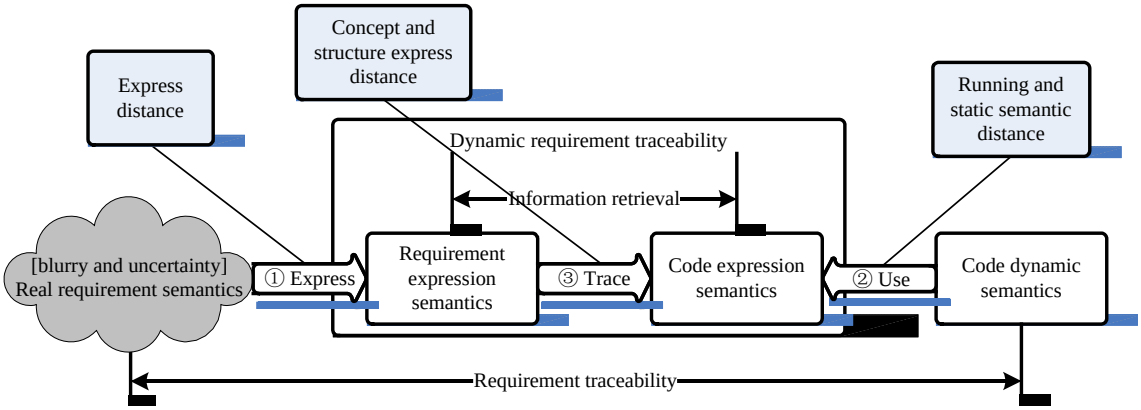


Fig.1 Traceability from requirement to code

图 1 需求到代码的跟踪关系

可以看出,手动建立需求与代码之间的跟踪关系,是通过理解需求与代码的语义,检查它们是否具有“实现”语义,但是由于机器无法理解语义信息,动态需求跟踪只能通过计算需求表达文字与代码表达文字之间的相似度来近似地建立需求与代码之间的跟踪关系.建立从需求到代码跟踪关系的误差主要来自 3 个方面:① 用户的期望和表达的误差.用户真正想要的是模糊的和不确定的,它可能随着时间的变化、环境的变换、高层策略的变化以及开发人员的反馈等而发生变化.将这种模糊的、不确定的需求语义用文字表达出来,必然存在偏差.② 运行功能与代码表达之间的误差.狭义地说,代码是软件开发的最终产物.代码的执行所表现出来的功能和代码的表现很可能不同,特别是在面向对象、面向构件等开发语言提出来之后,在部署/运行时确定代码的行为是相当常见的.③ 建立从需求表达语义到代码静态语义的跟踪关系时,还存在采用概念表示的差别和结构表达的差别.因此,采用 IR 建立动态需求跟踪关系存在大量的误差是可以理解的.

①中所产生的偏差是需求获取、需求表达、需求协商等活动需要解决的问题,这里不再讨论.程序的动态行为是无法采用 IR 技术建立与需求之间的关系的(②),文献[39,40]通过 TraceAnalyzer 记录运行中程序的足迹,以发现系统模块之间的跟踪关系,就是在静态的文本和执行的代码之间建立跟踪关系,可以作为 IR 技术的补充来使用.

即使忽略了①②所引入的误差,需求表达语义与代码静态语义之间建立跟踪关系也不是那么顺利的.需求是对问题领域的描述,而代码是解决领域的产物.从概念的表达上来说,需求和代码很可能采用不同的词来表达相同/相近的概念,而且同一个词也可以用来表示不同的概念.这就是造成精度问题最根本的原因.第 3 节将在精度问题分析的基础上,提出我们的解决思路.

3 动态需求跟踪框架

在对已有的研究进行分析、归纳的基础上,针对跟踪精度这个问题,我们提出了动态需求跟踪框架.该框架

主要由 3 部分组成:静态模型(static model)、执行模型(executive model)和维护模型(maintenance model).静态模型负责建立和更新需求跟踪关系;执行模型负责执行需求变更处理;维护模型在迭代开发生命周期中,负责提供更新需求跟踪关系的辅助信息.该框架以充分挖掘和利用开发活动中产生的辅助信息为核心,以改进传统的变更活动流程为手段,整合了变更流程中的几个活动,包括变更请求、影响分析、变更控制等,利用自动化工具挖掘和收集产生的辅助信息,从而能够提高动态需求跟踪建立和更新的精度.

动态需求跟踪框架主要采用两个方面的策略来提高跟踪精度.第一,针对不同的工作产品设计不同的跟踪关系计算方法.比如,对于建立需求到会议访谈的跟踪关系,因为会议访谈书写的形式很随意,利用一般的 IR 计算公式就可以.而对于代码来说,它所具有的层次结构能够辅助进行跟踪关系的选取.经过修正之后的相似度模型既能将部分错误的跟踪关系筛掉,又能引入漏掉的正确跟踪关系.第二,利用软件开发过程中积累的变更信息.现代大规模的软件开发不再是一蹴而就的,迭代的软件开发生命周期已经得到认同并广泛应用.需求和架构/设计之间往往是互相影响、互相补充、互相依赖的螺旋演化的关系[57].在这种环境下,需求跟踪也不是一成不变的,其建立和维护是随着系统的演化不断进行的.在需求发生变更时,根据需求跟踪关系进行影响分析和变更管理,对系统的演化进行控制;在需求发生变更之后,变更的内容又可以对需求跟踪关系进行更新和维护.系统演化的信息保存在数据库、需求管理工具、SCM 配置库等工具中,对这些工具中积累的变更进行挖掘,挖掘出来的信息可以自动地建立词之间的相似度信息,从而扩展现有的标引词相似度词典方法[29]为自动/半自动化的方法,可以用来指导下一阶段需求跟踪关系的更新和维护.

3.1 动态需求跟踪框架架构

图 2 所示为整个动态跟踪框架的 3 大部分:静态模型、执行模型和维护模型.静态模型针对跟踪关系的建立;执行模型针对跟踪关系的使用;维护模型对动态跟踪关系持续演化进行管理,为静态模型和执行模型提供支持.

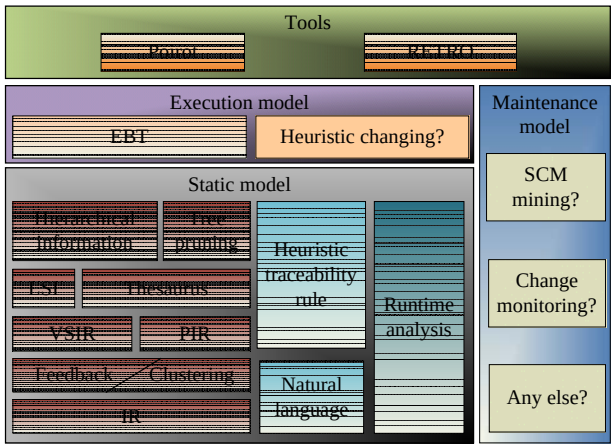


Fig.2 Architecture of dynamic requirement traceability

图 2 动态需求跟踪框架结构

静态模型对建立需求跟踪的技术和方法进行了刻画,基于 IR 的动态跟踪(第 1.1 节)、基于启发规则的动态需求跟踪(第 1.2.1 节)和基于运行时的分析(第 1.2.2 节)都属于静态模型,与传统的需求跟踪活动相对应.

执行模型以静态模型为基础,在需求发生变更时产生通知消息并进行记录,基于事件的跟踪(第 1.2.3 节)属于执行模型,与变更管理活动相对应.同时,我们认为执行模型中还可以加入启发式变更模块(heuristic changing),它可以在需求发生变更时,从支持信息库中获取以往的变更操作记录,从而提供给用户进行启发式的变更选择.

影响分析、变更管理等活动产生的信息可以用来辅助建立(更新)需求跟踪关系,对这些信息的挖掘和处理是通过动态需求跟踪框架中的“维护模型”进行的.维护模型主要包括两个模块的内容:对配置库/CASE 工具/数据库等信息的挖掘和对用户进行变更决策的监控.

3.2 跟踪精度增强策略

动态需求跟踪框架采用两种策略来支持建立和更新需求跟踪关系:利用细分之后的工作产品的特点和挖掘系统演化中的变更信息.

在已有的研究中,比较充分地利用了工作产品的特点方法都集中在基于 IR 的动态需求跟踪模型中,如图 2

中的静态模型.反馈(feedback)和聚类(clustering)都是 IR 中的常用技术.反馈的使用可以增强相关文献的响应而抑制非相关文献,它根据用户的理解来修正简单的语法匹配,因此,经过反馈之后产生的结果更加接近真实的跟踪关系.聚类的使用可以将具有相同属性的文档聚合成簇,这是从文档集合特有的属性挖掘出的信息.LSI 方法将基本的字词匹配用相似的语境匹配来代替,可以近似地认为语境的匹配相当于对语义的理解.标引词相似度词典通过手动创建词之间的相似度,引入相似系数并建立了相关性辞典,利用该辞典对 IR 方法中的相似度计算模型进行了修正,最后的跟踪关系的建立实际上包含了用户的语义理解.从实现原理上来看,LSI 与标引词相似度词典有相似之处,LSI 通过建立“词-文献”矩阵,隐式地获取了词之间的相关度,而标引词相似度词典方法是由分析人员手动建立的相关度,因此标引词相似度词典方法的查准率更高.但是,LSI 方法的优势在于其完全自动化的处理.层次结构信息和树剪枝操作对具有树形结构的需求和工作产品之间建立跟踪关系进行支持,利用结构信息提高了查全率和查准率.这是从空间的角度来挖掘辅助信息的策略.

另一个策略是从时间的角度来挖掘辅助信息,对应动态需求跟踪框架中的维护模型.对于一个开发过程规范的项目来说,配置库信息记录了系统演化的所有信息,包括会议记录、原始需求、需求规约(文本方式记录,如果是采用需求管理工具进行需求管理,则需要从需求管理工具中挖掘信息)、设计文档、UML 设计、代码文件等等.通过对同一个对象前后两个版本的分析,可以挖掘出概念的表达是如何进行变更的,这对基于语法的匹配是很重要的信息.比如,建立需求和代码之间的跟踪关系,最开始根据设计编写代码时,可以认为需求和代码采用相同词来描述同一概念的几率会很高.随着开发的进行,代码中一些概念的表示与需求的概念表示相差越来越远,如果在开发完成之时再从头开始建立跟踪关系,那么其中精度肯定会很低.但是,如果能够从代码的变更信息中挖掘出概念表达的变更规则,就可以为自动建立跟踪关系提供有力的支持.另一方面,在需求发生变更之后,由执行模型对用户进行通知,用户根据变更进行分析之后作出响应.这样的信息比从配置库挖掘出来的信息更加立体.它具有执行的语义,能够为以后发生类似的变更提供参考.如果配置库中挖掘出的信息是“as-is”,那么对执行模型监控获取的信息就是“will-do”.

3.3 跟踪精度增强迭代过程

动态需求跟踪关系精度的增强是一个以静态模型为中心迭代的过程,如图 3 所示.

整个流程从构建静态模型开始,自动产生候选跟踪关系.分析人员对候选的跟踪关系进行反馈和筛选,产生跟踪关系.在需求变更建议产生之后,执行模型首先进行影响分析,当影响度小于设定阈值时提交用户进行变更.当需求发生变更时,执行模型根据静态模型中确定的跟踪关系,获得受影响的工作产品集合,根据一定的策略通知分析人员.执行模型还可以维护模型积累的变更数据,以启发式方式向用户提供在当前变更环境下可以采用的变更方案.维护模型从两个方面获取数据,一个是第三方的 CASE 工具(比如 SCM、需求管理工具等)记录的信息,另一个是从执行模型获取的分析人员实际的变更操作信息.前一种信息辅助静态模型的重构和更新,第 2 种信息可以用来挖掘出需求变更的启发式规则,辅助分析人员进行需求变更处理.这两种信息并不是孤立的,它们可以互相补充、互相促进,并且它们在一定条件下可以相互转化.

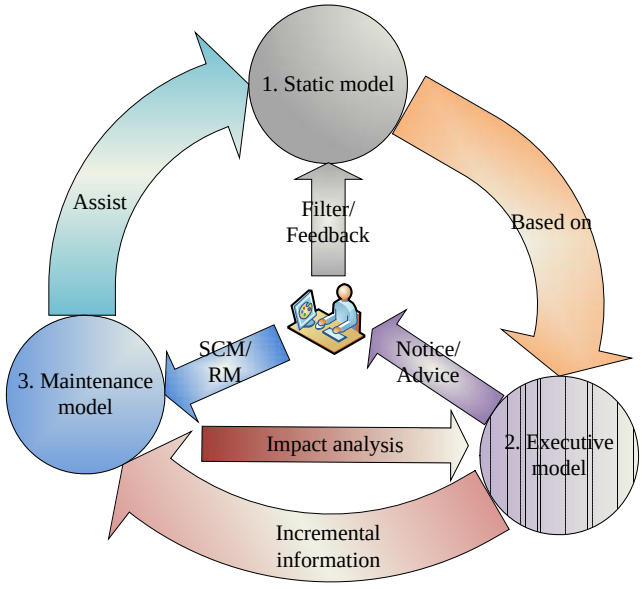


Fig.3 Iterative process of traces precision improvement

图 3 跟踪精度增强迭代过程

可以看出,需求跟踪关系的维护模型是静态模型和执行模型的支持模型,可以认为横切了静态模型和执行模型,记录静态模型和执行模型产生的信息,从中挖掘出分析人员的使用模式,并用这种使用模式进一步去指导静态模型的创建和执行模型的运行.在具有足够知识支持的情况下,它甚至可以代替人的大部分工作.

实际上,维护模型与知识库的作用类似.足够的知识加上合理的推理机制,我们认为可以近似代替对语义的理解.IR 技术是静态模型中最主要的技术手段.其推理逻辑比较简单,基于最简单的语法匹配.虽然语义相似和语法相似是两个层次上的问题,但我们认为,如果有足够的辅助信息,语法匹配就可以升级为语义匹配.所以,维护模型的加入可以在保持自动化程度的同时,提高静态模型的精度.

4 结论及展望

动态需求跟踪关系提供了自动化的方法来建立和维护需求跟踪,节省了时间,减少了错误.传统需求跟踪在实践中应用不广的问题得到了较好的解决,确认/验证、影响分析、变更管理等活动在组织管理有序的需求跟踪关系的辅助下能够更好地进行分析和决策,从而极大地降低了项目失败的风险.因此,需求跟踪作为 CMM/CMMI 2 级[5,6]和 ISO9001[6]中的必要活动,也不再因其是组织级的规定而被强迫使用,而是由于动态需求跟踪带来的自动化的优点,使其自然地成为项目不可或缺的活动之一.随着辅助工具的开发以及与已有 CASE 工具的集成,动态需求跟踪在项目实践中将会得到越来越多的使用,需求变更无序的情况将会得到极大的改善,为项目的最终成功打下坚实的基础.

本文对动态需求跟踪的研究现状进行了综述,包括基于 IR 技术的动态需求跟踪、基于跟踪规则启发的动态需求跟踪和基于事件的需求跟踪.在对目前动态需求跟踪面临的精度问题进行分析的基础上,提出了我们的解决方案——动态需求跟踪研究的框架.该框架包含静态模型、执行模型和维护模型 3 部分.它将传统的需求跟踪、影响分析、变更管理等活动进行了组合,利用该框架中“空间”上的工作产品的特点信息和“时间”上的系统演化变更信息来帮助提高动态需求跟踪的精度.

References:

[1] van Lamsweerde A. Requirements Engineering in the year 00: A research perspective. In: Proc. of the 22nd Int'l Conf. on Software Engineering (ICSE 2000). New York: ACM, 2000. 5−19.

[2] Jr Brooks FP. No silver bullet: Essence and accidents of software engineering. IEEE Computer, 1987,20(4):10−19.

[3] Nuseibeh B, Easterbrook S. Requirements engineering: A roadmap. In: Int'l Conf. on Software Engineering, Proc. of the Conf. on the Future of Software Engineering. New York: ACM, 2000. 35−46.

[4] Gotel OCZ, Finkelstein ACW. An analysis of the requirements traceability problem. In: Proc. of the 1st Int'l Conf. on Requirements Engineering Conf. (RE'94). IEEE Computer Society, 1994. 94−101.

[5] Gotel OCZ. Contribution structures for requirements traceability [Ph.D. Thesis]. London: Imperial College of Science, Technology and Medicine, University of London, 1995.

[6] Jarke M. Requirement tracing. Communications of the ACM, 1998,41(12):32−36.

[7] Ramesh B, Stubbs C, Powers T, Edwards M. Requirements traceability: Theory and practice. Annals of Software Engineering, 1997,3(1):397−415.

[8] Aurum A, Wohlin C. Engineering and Managing Software Requirements. New York: Springer-Verlag, 2005.

[9] Maletic A, Maletic JI. Recovering documentation-to-source-code traceability links using latent semantic indexing. In: Proc. of the 25th Int'l Conf. on Software Engineering. Washington: IEEE Computer Society, 2003. 125−135.

[10] Hayes JH, Dekhtyar A, Sundaram SK, Howard S. Helping analysts trace requirements: An objective look. In: Proc. of the 12th Int'l Requirements Engineering Conf. (RE 2004). Washington, 2004. 249−259.

[11] Hayes JH, Dekhtyar A, Sundaram SK. Advancing candidate link generation for requirements tracing: The study of methods. IEEE Trans. on Software Engineering, 2006,32(1):4−19.

[12] Cleland-Huang J, Settimi R, Duan C, Zou XC. Utilizing supporting evidence to improve dynamic requirements traceability. In: Proc. of the 13th IEEE Int'l Requirements Engineering Conf. (RE 2005). Washington: IEEE Computer Society, 2005. 135−144.

[13] Pohl K, Dömges R, Jarke M. Towards method—Driven trace capture. In: Proc. of the 9th Int'l Conf. on Advanced Information Systems Engineering. London: Springer-Verlag, 1997. 103−116.

[14] Spanoudakis G. Plausible and adaptive requirement traceability structures. In: Proc. of the 14th Int'l Conf. on Software Engineering and Knowledge Engineering. New York: ACM, 2002. 135−142.

[15] Robinson WN, Pawlowski SD, Volkov V. Requirement interaction management. ACM Computing Surveys (CSUR), 2003,35(2): 132−190.

[16] Ramesh B, Jarke M. Toward reference models for requirements traceability. IEEE Trans. on Software Engineering, 2001,27(1): 58−93.

[17] Ramesh B. Factors influencing requirements traceability practice. Communications of the ACM, 1998,41(12):37−44.

[18] Pohl K. Process-Centered Requirements Engineering. New York: John Wiley & Sons, 1997.

[19] Gotel O, Finkelstein A. Contribution structures. In: Proc. of the 2nd Int'l Symp. on Requirements Engineering (RE'95). IEEE Computer Society, 1995. 100−107.

[20] Jackson J. A keyphrase based traceability scheme. In: Proc. of the Tools and Techniques for Maintaining Traceability During Design, IEE Colloquium. London: IEEE Computer Society, 1991. 769−783.

[21] Kaindl H. The missing link in requirements engineering. ACM SIGSOFT Software Engineering Notes, 1993,18(2):30−39.

[22] Lefering M. An incremental integration tool between requirements engineering and programming in the large. In: Proc. of the IEEE Int'l Symp. on Requirements Engineering. IEEE Computer Society, 1993. 82−89.

[23] IBM. Rational RequisitePro. http://www-306.ibm.com/software/awdtools/reqpro/

[24] IBM. DOORS (dynamic object oriented requirements system). http://www.telelogic.com/index.cfm

[25] Borland. Borland CaliberRM. http://www.borland.com/us/products/caliber/index.html

[26] Change management & requirement tracing systems. http://www.bandwood.com/cms_exec_summary.htm

[27] Pinheiro FAC, Goguen JA. An object-oriented tool for tracing requirements. IEEE Software, 1996,13(2):52−64.

[28] Pohl K. PRO-ART: Enabling requirements pre-traceability. In: Proc. of the 2nd Int'l Requirements Engineering Conf. (RE'96). Washington: IEEE Computer Society, 1996. 76−84.

[29] Hayes JH, Dekhtyar A, Osbourne J. Improving requirements tracing via information retrieval. In: Proc. of the 11th Int'l Requirements Engineering Conf. (RE 2003). Washington: IEEE Computer Society, 2003. 151−161.

[30] Cleland-Huang J, Zemont G, Lukasik W. A heterogeneous solution for improving the return on investment of requirements traceability. In: Proc. of the 12th IEEE Int'l Requirements Engineering Conf. (RE 2004). Washington: IEEE Computer Society, 2004. 230−239.

[31] Dömges R, Pohl K. Adapting traceability environments to project specific needs. Communications of the ACM, 1998,41(12): 55−62.

[32] Antoniol G, Caprile B, Potrich A, Tonella P. Design-Code traceability for object oriented systems. Annals Software Engineering, 1999,9(1-4):35−58.

[33] Antoniol G, Canfora G, Casazza G, de Lucia AD, Merlo E. Recovering traceability links between code and documentation. IEEE Trans. on Software Engineering, 2002,28(10):970−983.

[34] Settimi R, Cleland-Huang J, Khadra OB, Mody J, Lukasik W, DePalma C. Supporting software evolution through dynamically retrieving traces to UML artifacts. In: Proc. of the 7th Int'l Workshop Principles of Software Evolution. Washington: IEEE Computer Society, 2004. 49−54.

[35] Cleland-Huang J, Settimi R, Benkhadra O, Berezhanskaya E, Christina S. Goal-Centric traceability for managing non-functional requirements. In: Proc. of the 27th Int'l Conf. on Software Engineering (ICSE 2005). New York: ACM, 2005. 362−371.

[36] Spanoudakis G, Zisman A, Pérez-Miñana E, Krause P. Rule-Based generation of requirements traceability relations. The Journal of Systems and Software, 2004,72(2):105−127.

[37] Cleland-Huang J, Chang CK, Ge YJ. Supporting event based traceability through high-level recognition of change events. In: Proc. of the 26th Int'l Computer Software and Applications Conf. on Prolonging Software Life: Development and Redevelopment (COMPSAC). Washington: IEEE Computer Society, 2002. 592−602.

[38] Cleland-Huang J, Chang CK, Christensen M. Event-Based traceability for managing evolutionary change. IEEE Trans. on Software Engineering, 2003,29(9):796−810.

[39] Egyed A. A scenario-driven approach to traceability. In: Proc. of the 23rd Int'l Conf. on Software Engineering (ICSE 2001). Washington: IEEE Computer Society, 2001. 123−132.

[40] Egyed A, Grüenbacher P. Automatic requirements traceability: Beyond the record and replay paradigm. In: Proc. of the 17th IEEE Int'l Conf. on Automated Software Engineering. Washington: IEEE Computer Society, 2002. 163−171.

[41] Sun JJ, Cheng Y, Ding Q, Li JJ, Song LL, Ke Q. Information Retrieval. Beijing: Science Press, 2004 (in Chinese).

[42] Deerwester S, Dumais ST, Furnas GW, Landauer TK, Harshman R. Indexing by latent semantic analysis. Journal of the Society for Information Science, 1990,41(6):391−407.

[43] Landauer TK, Foltz PW, Laham D. An introduction to latent semantic analysis. Discourse Processes, 1999,25:259−284.

[44] Baeza-Yates R, Ribeiro-Neto B. Modern Information Retrieval. Addison Wesley, 1999.

[45] Zou XC, Settimi R, Cleland-Huang J. Phrasing in dynamic requirements trace retrieval. In: Proc. of the 30th Annual Int'l Computer Software and Applications Conf. (COMPSAC 2006). Washington: IEEE Computer Society, 2006. 265−272.

[46] Cysneiros LM, Leite JCSP. Nonfunctional requirements: From elicitation to conceptual models. IEEE Trans. on Software Engineering, 2004,30(5):328−350.

[47] Gross D, Yu E. From non-functional requirements to design through patterns. Requirements Engineering Journal, 2001,6(1):18−36.

[48] Chung L, Nixon BA, Yu E, Mylopoulos J. Non-Functional Requirements in Software Engineering. Springer-Verlag, 1999.

[49] Cleland-Huang J, Schmelzer D. Dynamically tracing non-functional requirements through design pattern invariants. In: Proc. of the Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE 2003). 2003.

[50] Cleland-Huang J. Requirements traceability—When and how does it deliver more than it costs? In: Proc. of the 14th IEEE Int'l Requirements Engineering Conf. (RE 2006). Washington: IEEE Computer Society, 2006. 323.

[51] Hayes JH, Dekhtyar A. A framework for comparing requirements tracing experiments. Int'l Journal Software Engineering and Knowledge Engineering (IJSEKE), 2005,15(5):751−781.

[52] IBM. Purifyplus. http://www-304.ibm.com/jct03001c/software/awdtools/purifyplus/

[53] Lin J, Lin CC, Cleland-Huang J, Settimi R, Amaya J, Bedford G, Berenbach B, Khadra OB, Duan C, Zou XC. Poirot: A distributed tool supporting enterprise-wide automated traceability. In: Proc. of the 14th Int'l Requirements Engineering Conf. (RE 2006). Washington: IEEE Computer Society, 2006. 356−357.

[54] Hayes JH, Dekhtyar A, Sundaram SK, Holbrook EA, Vadlamudi S, April A. Requirements tracing on target (RETRO): Improving software maintenance through traceability recovery. Innovations System Software Engineering, 2007(3):193−202.

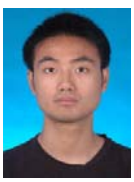
[55] Li Y, Li J, Yang Y, Li MS. Requirement-Centric traceability for change impact analysis: A case study. In: Wang Q, Pfaho D, Raffo DM, eds. Proc. of the Int'l Conf. on Software Process (ICSP 2008). Berlin, Heidelberg: Springer-Verlag, 2008. 100−111.

[56] Princeton University. WordNet. http://wordnet.princeton.edu/

[57] Nuseibeh B. Weaving together requirements and architectures. Computer, 2001,34(3):115−117.

附中文参考文献:

[41] 孙建军,成颖,丁芹,李君君,宋玲丽,柯青.信息检索技术.北京:科学出版社,2004.



李引(1981－),男,重庆人,博士生,主要研究领域为需求工程.

李娟(1977－),女,博士,副研究员,CCF 会员,主要研究领域为需求工程,过程改进.

李明树(1966－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为软件工程方法学,软件过程技术,需求工程,软件工程经济学.